1972
international book year

KFKI-72-32

Krammer Gergely

Lovas Istvánné

SLIP

SZIMMETRIKUS LISTAKEZELŐ SZUBRUTIN RENDSZER AZ MTA CDC 3300-AS ÉS ICT 1905-ÖS SZÁMOLÓGÉPÉN

Hungarian Academy of Sciences

CENTRAL
RESEARCH
INSTITUTE FOR
PHYSICS

BUDAPEST

KFKI-72-32

SLIP

- SZIMMETRIKUS LISTAKEZELŐ SZUBRUTIN RENDSZER AZ MTA CDC 3300-AS ÉS ICT 1905-ÖS SZÁMOLÓGÉPÉN

Krammer Gergely (MTA Automatizálási Kutató Intézet)
Lovas Istvánné (MTA Központi Fizikai Kutató Intézet)

Computing Techniques Department

ABSTRACT

SLIP is a list processing system in which each element is linked to both its predecessor and its successor. The system is embedded in the FORTRAN language as a set of subroutines. SLIP was developed by J.Weizenbaum. The system has been implemented at ICL 1905 and CDC 3300 computers in order to provide possibilities for list processing and symbolic manipulation of algebraic expressions.

KIVONAT

A SLIP szimmetrikus listák kezelésére szolgáló FORTRAN szubrutin rendszer. Weizenbaum dolgozta ki 1963-ban. A rendszer implementációja az ICL 1905-ös és a CDC 3300-as számológépekre készült el, abból a célból, hogy megteremtse a listafeldolgozás és az algebrai kifejezésekkel végzett müveletek lehetőségét.

РЕЗЮМЕ

Для обработки симметричных списков Вейзенбаумом в 1963 г. была разработана система подпрограмм "SLIP", написанная на языке FORTRAN. С целью обработки списков и проведения операций с алгебраическими выражениями система была приспособлена к ЭВМ ICL 1905 и CDC 3300.

TARTALOMJEGYZÉK

1. BEVEZETÉS

A SLIP szimmetrikus listák kezelésére szolgáló FORTRAN szubrutin rendszer. Weizenbaum 1963-ban közölte az IBM 7090-re kidolgozott szubrutinokat ([1]) és azóta több más géptipusra adaptálták azokat.

A lista a számitógépek nem numerikus alkalmazásainak nagy részénél jól használható fogalom. A lista elemekből áll; a lista elemei vagy tovább nem bontható adatok: atomok, vagy listák: a főlista allistái. Az atomok jelentése alkalmazásonként változó, hasonlóképpen a listák fenti alaptulajdonságon tulmenő sajátságai.

A listák tárolására a jelenleg elterjedt, szekvenciális cimzésü számológépmemóriákban szomszédos memóriaszavakból kialakitott cellákat használunk. A szimmetrikus listák elemei olyan cellák, amelyek tartalmaz/hat/nak atomi információt és tartalmazzák a lista következő elemének és megelőző elemének gépi cimét. Ezeket a mutatókat /pointereket/ követve a lista egymás utáni elemei könnyen kezelhetők, a lista bővithető, lebontható.

A lista elemei mutatókkal vannak egymáshoz kapcsolva, nem kell, hogy a memóriában egymás utáni szavakban helyezkedjenek el. Olyan alkalmazásoknál, amelyeknél a program futása során a listák dinamikusan változnak, ez a szervezés lehetőséget nyujt a rendelkezésre álló memória jó felhasználására: a pillanatnyilag szabad cellák egy közös cellaláncon találhatók, amelyről minden lista igényének megfelelően kaphat ujabb cellákat.

x x x

Ebben a leirásban a lehetséges alkalmazásokra csak az irodalomjegyzéken keresztül utalunk. Célunk az MTA két számológépére párhuzamosan meghonositott SLIP szubrutinok felhasználói szintü leirása. A rendszer nagyobb része FORTRAN-ban irt szubrutinokból áll, amelyek más géptipuson is használhatóak, ezen felül néhány assembly nyelven irt szubrutint is tartalmaz, amelyeket más géptipusokra külön meg kell irni.

A 2-4. részekben vázlatosan leirjuk a rendszer használatát, majd az 5. részben a rendszer valamennyi szubrutinját. Végül a 6. rész-

ben példákat mutatunk be. A 7. rész tartalmazza az MTA CDC 3300-as és ICL 1905-ös számológépén implementált rendszer felhasználásával kapcsolatos tudnivalókat.

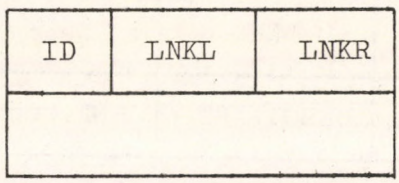
Munkánk célja egy, a nem numerikus alkalmazásokban jól használható rendszer megvalósitása volt, a tudományos kutatómunka elősegitése érdekében. Adott célra természetesen kialakitható ennél megfelelőbb rendszer is, de előzőleg a SLIP-en tanulmányozhatók az ilyen rendszerek tulajdonságai.

Munkánkban nagy segitséget nyujtottak a felmerülő problémák megvitatásában Ivanyos Lajosné, Tarján Mihály és Zimányi Józsefné tudományos munkatársak; ezuton mondunk köszönetet.

2. A SLIP ADATSTRUKTURA

Az alap információs modul a SLIP-ben az ugynevezett cella, amely két, a memóriában egymás után következő "tárolóegységből" áll.

A cella első részét CONT résznek nevezzük, a második rész a DATUM rész, amely a felhasználó szempontjából az atomi információ hordozója:



A CONT rész további három részre bomlik: ID, LNKL és LNKR.

Az ID rész tartalma:

0 - ha a DATUM rész a felhasználó által értelmezett adat,
1 - ha a DATUM rész egy lista "neve",
2 - a cella egy lista feje,
3 - a cella egy lista "reader"-je.

Az LNKL és LNKR mező az előző /bal/, illetve a következő /jobb/ cella gépi címét tartalmazza. A readerek kivételével minden cella egy és csak egy listához tartozhat, vagy egy a felhasználó által készitett listához, vagy a rendszer által kezelt listához, az ugynevezett "szabad cellák listájá"-hoz.

Ha a cella ID része 0, a DATUM része bármilyen információt tartalmazhat. Lehet az adat egész vagy lebegőpontos szám, karakter stb., az adat előállitásáról és felhasználásáról a programozó által irt FORTRAN program gondoskodik. Az alap strukturális egység a SLIP-ben a lista. A lista egy vagy több összekapcsolt cellából áll, a cellák az LNKL és LNKR mezőkön - a szomszédos cellák gépi cimén - keresztül kapcsolódnak egymáshoz. Minden listaelem /cella/ cime előfordul a szomszédos listaelem LNKL mezőjében és a másik szomszéd LNKR mezőjében.

Minden listának van egy kitüntetett eleme /cellája/, amelynek az ID része = 2. Ezt a cellát a lista fejének nevezzük. Minden listának csak egy feje lehet. Mivel minden listának van egy feje, ezen keresztül a listára való hivatkozás egyértelmü. A listára való hivatkozás egy listanévvel történik; a listanév olyan szó, mely tartalmazza a listafej gépi cimét. Ez a szó lehet a program valamely változója, vagy szerepelhet egy ID = 1 tipusu cella DATUM részében.

A listafej CONT részének LNKL-je a lista legalsó elemére, az LNKR-je a lista legfelső elemére mutat.

A listafej DATUM része ugyanolyan mezőkre van felosztva, mint a CONT része:

2	LNKL	LNKR
MK	DESC.LIST	REF.COUNT

MK /LIST MARK/ mező a lista megjelölésére szolgálhat,
DESCR.LIST mezőben vagy a leiró lista cime van, vagy 0, /a leiró lista fogalmának magyarázatát később megtaláljuk/,
REF.COUNT mező egy egész számot tartalmaz, mely megmutatja, hogy ez a lista hányszor fordul elő allistaként.

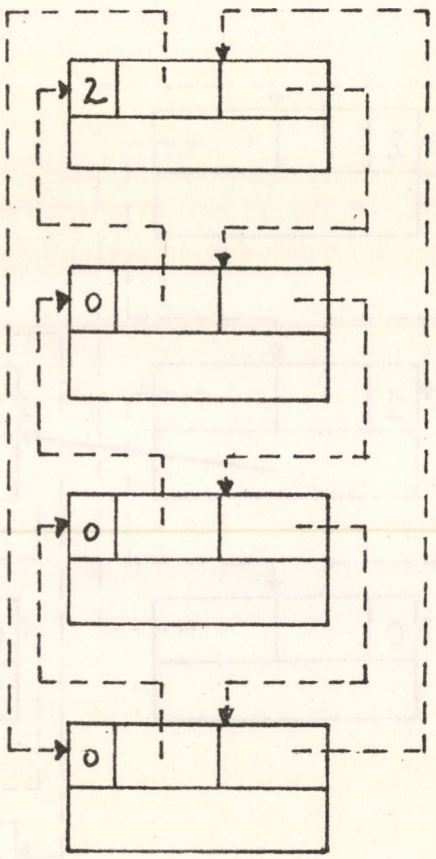
Az L1 listáról azt mondjuk, hogy az L2 lista allistája, ha az L1 lista nevét az L2 lista egy eleme a DATUM részében tartalmazza. A nevet tartalmazó cella ID része = 1, az ilyen cellát listakapcsolónak nevezzük.

1	LNKL	LNKR
LISTANEV		

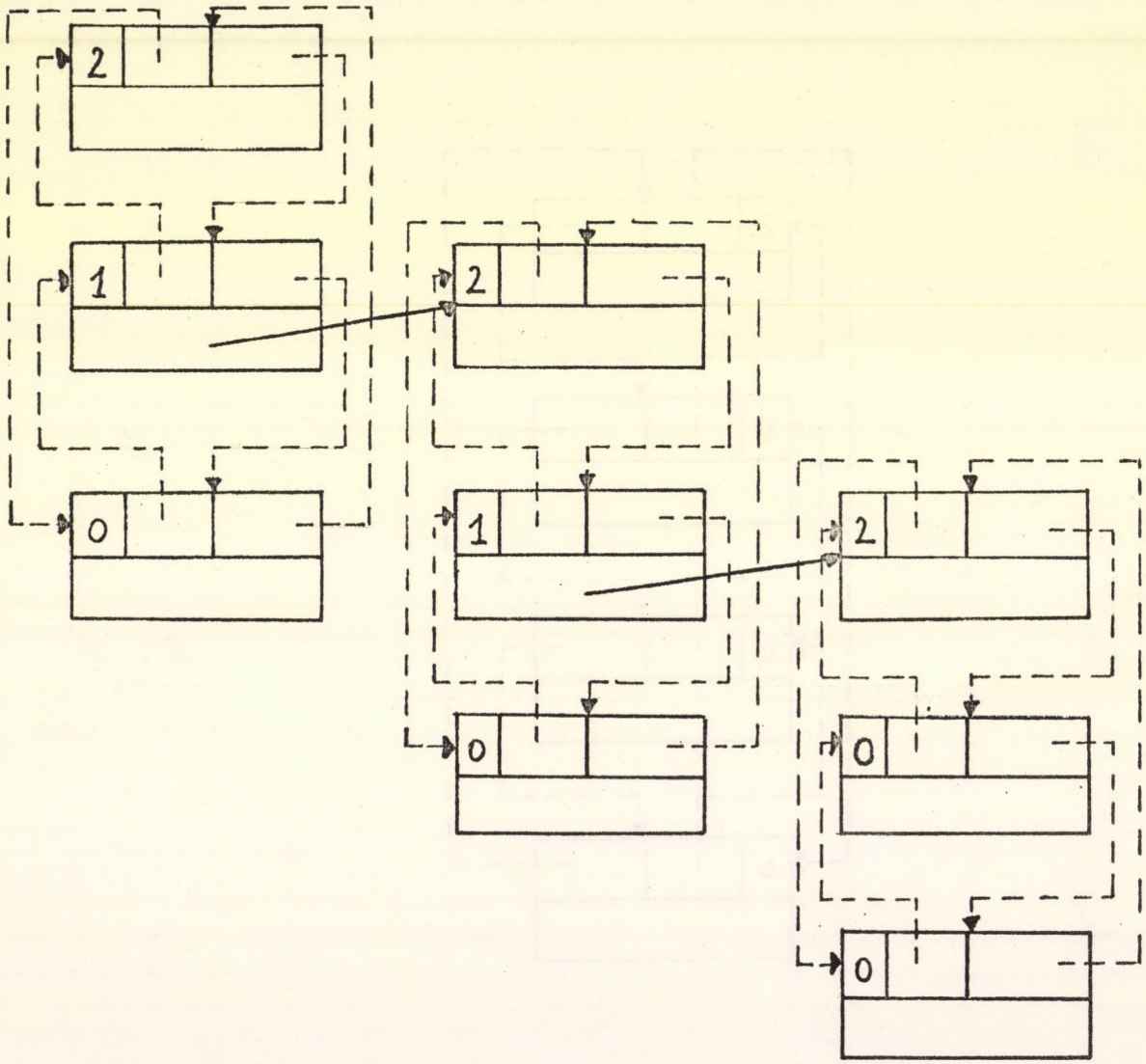
Nincs alapvető különbség lista és allista között, bármely lista előfordulhat egy másik lista allistájaként.

A legbonyolultabb szerkezeti egység a listastruktura, amely egy fő listából és tetszőleges számu allistából állhat. Az allisták lehetnek a fő lista allistái vagy egy allista allistái, stb. Ilyen összetett listastrukturával ábrázolható például egy fastruktura.

Közös allisták megengedettek; több lista közös allistája csak egyszer fordul elő a memóriában.



Példa egyszerü listára



Példa összetett listára

3. PROGRAMSTRUKTURA

A SLIP-rendszer alapját az ugynevezett bázis primitiv rutinok alkotják; ezek végzik a rendszer alapvető funkcióit. Megadott értékeket betöltenek egy SLIP-cella valamelyik /ID, CONT, LNKL, LNKR, DATUM/ mezejébe, illetve függvényértékként szolgáltatják a cella valamelyik mezejében tárolt értéket.

A bázis primitiv rutinok általában a gépi szó részein végeznek müveletet, megirásukhoz ismerni kell a cella szerkezetét, az adott gépen való konkrét megvalósitását, ezért ezek a rutinok assembly nyelven iródtak.

A felhasználó általában a bázis primitiv rutinokat közvetlenül nem használja; a SLIP-rendszer magasabb szintü rutinjai azonban ezekre a rutinokra épülnek, mint alapra. A bázis primitivektől eltekintve a SLIP-rendszer rutinjai FORTRAN nyelven iródtak. A FORTRAN rutinok jól elhatárolható csoportokat alkotnak, amelyek mindegyike a listakezelés egy-egy módjára vonatkozik. A legtöbb SLIP-rutin FORTRAN-függvényként van megirva, de nincs logikai különbség a között, hogy az illető rutint függvényként vagy szubrutinként aktivizáljuk. Egy FORTRAN--modulon belül azonban a legtöbb számológép FORTRAN-reprezentációja csak az egyik módot engedi meg.

Például a

```
J = LIST (K)
```

utasitással éppugy létrehozhatunk egy listát, mint - egy másik FORTRAN-modulban - a

```
CALL LIST (K)
```

utasitással.

A FORTRAN-ban a változóknak és függvényeknek tipusa van ; ha más tipusdeklarációt nem adunk, az I, I, K, L, M, N betüvel kezdődő azonositók egész tipusuak, a többi valós tipusu. A SLIP-rutinok ragaszkodnak ehhez az implicit tipusdeklarációhoz.

A felhasználónak tehát tekintetbe kell vennie a függvények ti-

pusát, amikor felhasználja őket: például a SEQLL*-függvény az implicit tipusdeklaráció miatt valós. A cella DATUM-részét a SEQLL-függvény segitségével csak egy

A = SEQLL (S,F)

tipusu utasitással tudjuk változatlan formában átvenni, a

K = SEQLL (S,F)

utasitás esetében lebegőpontos-fixpontos konverzió történik.

Az atomi információ kezelése teljesen a felhasználó feladata, igy a rendszer a DATUM-rész tipusát sem ismeri. Igy ez a tipuskonverzió a tartalom tulajdonképpeni tipusától /egész, valós, logikai, karakter, cim stb./ független, formális konverzió.

A nem kivánatos konverzió elkerülésére több lehetőség van. Például:

```
      ...
      EQUIVALENCE (INTREG, VALREG)
      ...
      ...
      VALREG = SEQLL (S, F)
C     INTREG-BEN BENNE VAN VALTOZATLAN FORMABAN A
C     CELLA DATUM RESZE
      ...
```

A rendszerhez tartozik az INTGER-függvény, amely lehetővé teszi, hogy egy valós tipusu változóban elhelyezett értéket változtatás /konverzió/ nélkül áttöltsünk egy egész tipusu változóba. Például:

K = INTGER (SEQLL (S, F))

A SLIP teljes mértékben kihasználja a FORTRAN-nak azt a tulajdonságát, hogy a függvényhivások többszörös mélységben egymásba skatulyázhatók.

---

* A nagybetüs röviditések egy-egy SLIP rutin nevét jelentik, amelynek pontos leirását a az 5. fejezetben talaljuk meg.

4. A SLIP-RUTINOK FŐBB CSOPORTJAI

4.1 A rendszer előkészitése és elinditása

A legtöbb listakezelő rendszerhez hasonlóan a SLIP a memóriát dinamikusan jelöli ki. Ez azt jelenti, hogy a rendszer elkészíti a "szabad cellák listáját" /LIST OF AVAILABLE SPACE, a továbbiakban LAVS/. s valahányszor egy uj cellára van szükség, leakaszt egy cellát a LAVS-ról. Ha valamely cellára a továbbiakban már nincs szükség, a rendszer a felszabadult cellát visszaakasztja a LAVS-ra.

A LAVS-t az INITAS rutin készíti el a felhasználó által erre a célra kijelölt tömbből. A felhasználónak a programjában deklarálni kell egy egydimenziós tömböt és hivnia kell az INITAS szubrutint. Tehát minden programnak a következőképpen kell kezdődnie:

```
PROGRAM PELDA
DIMENSION AREA (5000)
.
.
.
CALL INITAS (AREA, 5000)
.
.
.
```

A továbbiakban a felhasználó közvetlenül nem hivatkozhat az AREA tömbre. 5000 helyett természetesen más egész szám is állhat a DIMENSION utasitásban.

A "szabad cellák listája" formáját tekintve tulajdonképpen nem SLIP-lista, mivel a kapcsolat a lista-elemek között csak egy irányban van meg.

4.2 Helyigénylés és felszabaditás

Bármilyen dinamikus helyigénylő rendszerben a memória eredményes kihasználása azon alapul, hogy azt a helyet, melyre nincs tovább szükség, azonnal elengedjük. Felmerül a kérdés, ki határozza meg, hogy mikor lehet egy memóriaegységet elengedni? Vannak nyelvek, /pl.: LISP, SNOBOL/, melyekben a memória feletti rendelkezés teljesen a rendszer-

hez tartozik, a rendszer szisztematikusan végigvizsgálja a memória-egységeket, és összegyüjti azokat, amelyekre nincs hivatkozás /"garbage collection"/. Más nyelvekben /pl.: IPL-V/ ez teljesen a programozó feladata. A SLIP-ben a memória feletti rendelkezés felelőssége megoszlik a rendszer és a felhasználó között. A listák megőrzésének, illetve megsemmisitésének alapja az ugyenevezett referencia count. A listafejben van egy mező, amelyben tárolt szám megmutatja, hogy az illető lista hány lista allistája. A rendszer automatikusan megnöveli 1-gyel a referencia count-ját annak a listának, melynek a nevét egy másik listán elhelyeztük, illetve csökkenti a referencia count-ot, ha a lista nevét kitöröljük egy listáról, vagy ha a nevet tartalmazó listát magát kitöröljük.Amikor egy lista referencia countja nullára csökkent, erre a listára nincs több hivatkozás, a rendszer a listát elengedi, azaz a lista celláit visszahelyezi a LAVS-re. Mindaddig, mig a referencia count pozitiv, a lista aktiv használatban van, a listát a rendszer megőrzi.

Bár az automatikus ellenőrzés általában megfelelő, a felhasználó időnként mégis maga akarja az ellenőrzést irányitani. /Pl.: egy lista egy adott pillanatban nincs használatban, később azonban szükség lesz rá./ A felhasználónak megvan a lehetősége, hogy egy lista kezdeti referencia countját 0-ra vagy 1-re állitsa be. Ha 0-ra állitotta be, a lista elengedése automatikus lesz, ha 1-re, a referencia count sohasem lehet 0, tehát az illető listát csak a felhasználó engedheti el az erre szolgáló rutin /IRALST/ segitségével. A felhasználó által kezelt lista nevét természetesen egy változóban meg kell őrizni.

A rendszer minden listán elhelyezendő adatot megvizsgál, hogy az adat nem listanév-e. Ha az adat listanév, ez azt jelenti, hogy az adott nevü listát allistává kivánjuk tenni, tehát a referencia countját 1-gyel meg kell növelni.

Amikor egy uj cellára van szükség, pl.: uj adatot kivánunk a listán elhelyezni, a rendszer elvesz egy cellát a LAVS tetejéről. Amikor egy cellára nincs tovább szükség, a rendszer a felszabadult cellát visszaakasztja a LAVS aljára. A SLIP szimmetriájának egy érdekes következménye, hogy egy egész listát egyszerre el lehet engedni, azaz a LAVS-ra visszaakasztani, anélkül, hogy magukkal a cellákkal bármit is csinálnánk, mindössze 3 mutatót kell kicserélni a lista hosszától függetlenül. A rendszer a cella elengedésének időpontjában nem foglalkozik annak vizsgálatával, hogy a cella listakapcsoló-e, s igy a megfelelő allista referencia countjának csökkentését sem végzi el. Erre uj cella igénylésekor kerül sor. Amikor a rendszer levesz a LAVS-ról egy cellát,

mindig megvizsgálja, hogy nem tartalmaz-e listanevet, ha igen, a megfelelő referencia countot csökkenti eggyel. Ha az uj referencia count 0, a rendszer elengedi az egész listát.

Tehát egy olyan allista (tényleges) elengedése, amelyre már sehonnan nincs hivatkozás, akkor történik csak meg, ha az általa lefoglalt cellákra ujból szükség van. Ezzel a rendszer időt takarit meg.

4.3 Listakészités. Információ elhelyezése és visszanyerése

Listát a legegyszerübb módon a LIST-függvény hivásával készithetjük. Az

L = LIST (K)

utasitás hatására elkészül egy üres listafej. Ha az input paraméter 9 /LIST (9)/, az elkészült lista referencia countja 0 lesz, különben a referencia count értéke 1.

A felhasználó a listára a lista nevével hivatkozhat. A név egy FORTRAN tárolóegység, amely tartalmazza a listafej gépi cimét. A listafej cimének tárolási módja a névben ellenőrzést is jelent; ha egy listára a nevével hivatkozunk, a rendszer ellenőrzi, hogy valóban név-e, a benne lévő cim reális-e és a cim valóban egy listafejre mutat-e.

A LIST-függvény híváskor a függvény értékeként a lista nevét adja eredményül. Ha az input paraméter nem 9, hanem egy K változó /amelynek értéke nem 9/, akkor K-ban is a lista nevét kapjuk.

Rendelkezésre állnak rutinok az adatok listán való elhelyezésére és visszanyerésére. A SLIP szimmetriájából következik, hogy egy adat elhelyezése a listán mindig uj elem beszurását jelenti, még akkor is, ha a lista üres. Elhelyezhetünk egy uj adatot, egy adott listaelem után jobbra, vagy az adott listaelem elé balra /NXTRGT, NXTLFT/. Természetesen az adott listaelem lehet maga a listafej is. Ilyenkor azt mondjuk, hogy a lista tetejére ill. aljára helyeztük el az adatot /NEWTOP, NEWBOT/. Kicserélhetjük egy adott listaelem adatrészét /SUBST/, vagy a lista tetején, illetve alján lévő adatot /SUBSTP, SUBSBT/.

Érdekes tulajdonsága a SLIP-rendszernek, hogy a lista végét /BOT-ját/ ugyanolyan könnyen elérhetjük, mint a lista elejét /TOP-ját/, függetlenül a lista hosszától. Megkaphatjuk a lista tetején, illetve alján tárolt adatokat a TOP, BOT függvények hivásával, kitörölhetünk adatot a lista tetejéről, illetve aljáról /POPTOP, POPBOT/.

4.4 Végighaladás a listán, a SEQUENCER és a READER

A SLIP-rendszer előnye abban mutatkozik meg, hogy a felhasználónak a feldolgozáskor nem kell mindent tudnia a lista strukturájáról. Tudnia kell, hogy mit kell tennie a következő listaelemmel, de nem kell tudnia, hogy hány elem van a listán, vagy hány allista kapcsolódik a főlistához stb. Az indexelési technika gondját a rendszer leveszi a programozó válláról.

Az S = SEQRDR (L) utasitással kell kezdeni a listán való végighaladást; és S lesz a végighaladást irányitó "sequencer".

A listafeldolgozás lehet szekvenciális. Meg kell határozni a feldolgozás irányát /jobbra vagy balra haladunk-e a listán/ és módját: lineáris-e a feldolgozás, azaz csak a fő listát kivánjuk feldolgozni, vagy strukturális: a feldolgozást az allistákra is ki kivánjuk terjeszteni.

A sequencer /vagy sequence reader/ lényegében egy mutató, mely a soronkövetkező listaelem /cella/ cimét tartalmazza. Ha a feldolgozásnál a listán továbbhaladtunk, a rendszer automatikusan megváltoztatja a sequencer tartalmát, mindig az aktuális cella cimét irja be a sequencerbe. A sequencer egy FORTRAN-változó, nem a LAVS-ból származó cella!

A szekvenciális mechanizmus nem őrzi meg a feldolgozás "történelmét", azaz, ha egyszer leszálltunk egy szublistára, onnan a főlistára visszatérni nem lehet.

Egy listának több sequencere is lehet egyszerre. Kezdetben a sequencer a lista fejére mutat.

Ahhoz, hogy meghatározzuk a következő cellát, meg kell mondanunk a haladás irányát. Ezért mind a lineáris, mind a strukturális feldolgozásnál két függvény áll rendelkezésre a jobb irány és a bal irány számára. A SEQLR/SEQLL függvények az ugynevezett lineáris sequencer függvények.

Az ADAT = SEQLR (S,F) utasitás hatására az S sequencer a soronkövetkező cella cimét fogja tartalmazni. Az F FORTRAN változó információt tartalmaz arra vonatkozóan, hogy az aktuális cella egy listafej, listakapcsoló vagy listaelem-e. A függvény értéke a cella DATUM része.

A SEQSR/SEQSL függvények abban különböznek az előzőktől, hogy az előrehaladás strukturális. Ha a feldolgozás során egy listakapcsoló-

hoz érkezünk, az allistába való leszállás automatikusan megtörténik.

A feldolgozás során gyakran nemcsak le kell szállni a legalacsonyabb szintü allistába, hanem vissza is kell térni a főlistára. Az indexelés azonban lista-struktura esetén sokkal bonyolultabb, mint egy tömb esetén, hiszen az indexelés "történelmét" is meg kell őrizni ahhoz, hogy a visszatérés a főlistába megtörténhessék.

A reader-mechanizmus ezt a nehézséget megoldó indexelési technika.

A reader olyan mint egy verem, amelybe a mutatók egymás fölé vannak eltemetve, egy adott pillanatban csak a legfelső "él" közülük, a többi az indexelés történelmét őrzi /push-down stack/.

A reader a LAVS-ról leakasztott cellákból áll. A readerverem legfelső elemének cime - a reader neve.

Egy readercella a következő információt tartalmazza:

3	LPNTR	link
	LOFRDR	LCNTR

LPNTR tartalmazza a lista aktuális elemének cimét,
LOFRDR az aktuális lista fejének a cime,
LCNTR a level counter, mely megmutatja, hogy milyen szintü allistában vagyunk,
link a veremben lévő következő readercella cime.

A K = LRDROV (L) utasitás létrehozza az L lista K nevü readerjét, azaz létrehoz egy cellát, amelyben kezdetben az LPNTR és LOFRDR mező az L nevü lista fejének cimét tartalmazza, LCNTR és a link 0.

Az advancer rutinok hasonlóak a sequencer rutinokhoz, de bővebbek abban az értelemben, hogy a felhasználó definiálhatja a keresés tárgyát /milyen adatot keres/. A keresés tárgya lehet:

- egy listaelem /ID = 0/
- egy listanév /ID = 1/
- fejtől különböző listaelem /ID = 0,1/

A keresés módjának /lineáris vagy strukturális/, irányának /jobb vagy bal/ és tárgyának /elem, kapcsoló, szó/ figyelembevételével összesen 12 lehetséges advancer rutin lehet.

Egy advancer rutin neve tartalmazza a keresés módját, tárgyát és irányát.

Az ADV xyz (K, F) advancer rutin nevében

x a keresés módját határozza meg:
- L lineáris
- S strukturális

y a keresés tárgyát határozza meg
- E elem
- N listanév
- W fejtől különböző listaelem

z a keresés irányát határozza meg
- L bal irány
- R jobb irány

Például az

D = ADVLEL (K, F)

utasitás hatására a K reader a listán bal irányban, lineárisan halad és listaelemet keres. A K reader LPNTR-je mindig az aktuális listaelem cimét tartalmazza. Allistákba nem történik leszállás. Ha a megadott tárgyat megtalálta /jelen esetben egy listaelem/, az F FORTRAN változó értéke 0. Ha az előrehaladás során a tárgy megtalálása előtt a listafejhez érkezik, F nullától különböző értéket vesz fel. A függvény értéke a megtalált tárgycella DATUM része.

A strukturális advancerek abban különböznek a lineáris advancerektől, hogy ha a reader az előrehaladás során allistához érkezik, a kutatást az illető tárgycella után az allistán folytatja.

Ha a strukturális feldolgozás során a reader egy listakapcsolóhoz érkezik, a rendszer leakaszt egy uj cellát a LAVS-ról és hozzákapcsolja a reader-cellához. Ez az uj cella fogja tartalmazni az információt a feldolgozás eddigi menetéről, az eredeti readercella /melynek cimét a K = LRDROV (L) utasitás szolgáltatta/ pillanatnyi tartalmát. Az eredeti readercella az allista adatait tartalmazza: LOFRDR-je most az allista fejének cimét fogja tartalmazni, LPNTR része az ezen allistán aktuális elem cimét, a link része pedig az uj cella cimét. LCNTR értéke

1-gyel megnövekszik, jelezvén, hogy 1-gyel mélyebb szintü allistára történt leszállás.

Ha az allista minden elemét feldolgoztuk /visszaérkeztünk a fejhez/, a rendszer visszaállitja a readercella tartalmát az ezt megelőző állapot LOFRDR és LPNTR értékek válnak aktuálissá és a readerlánc megfelelő elemét /mely ezeket a LOFRDRR és LPNTR értékeket tartalmazta/ elengedi. Az LCNTR értékét 1-gyel csökkenti.

Például az

D = ADVSEL (K, F)

strukturális advancer a feldolgozás során leszáll az allistákba. K a reader neve. Az F FORTRAN változó értéke 0, ha a megadott tárgyat megtalálta, és 0-tól különböző, ha a főlista fejéhez visszaérkezett. Ha a reader a főlista fejét megtalálta, a kutatás abbamarad, ha azonban egy allista fejéhez érkezik, a reader felemelkedik a magasabb szintü listára és a kutatás ezen a listán folytatódik. A függvény értéke a megtalált tárgycella DATUM része.

4.5 A leiró lista

Minden listához hozzákapcsolhatunk egy ugynevezett leiró listát. A leiró lista cellapárokból áll, a cellapár első eleme az ugynevezett attributumot tartalmazza, a második elem az attributumhoz tartozó értéket.

A leiró lista nem allista, mert a leiró lista neve nem jelenik meg semmilyen listakapcsolóban. Az L lista leiró listájának cimét az L lista feje tartalmazza. A leiró listával foglalkozó rutinok paraméterként az L lista nevét kapják meg.

Az LL listát az L leiró listájává tehetjük a MAKEDL (L, LL) szubrutin-hivással. Az N = NEWVAL (AT, VAL, L) utasitás az L lista leiró listáján megkeresi az AT attributumot és a hozzátartozó értéket VAL-ra cseréli ki. Ha AT attributum nem volt az L lista leiró listáján, AT uj attributumként felkerül a leiró listára a hozzátartozó VAL értékkel együtt. Megkereshetünk egy adott attributumhoz tartozó értéket az ITSVAL (AT, L) rutin segitségével stb.

4.6 Rekurzió

A FORTRAN nem rekurziv nyelv, a SLIP-rendszerben azonban van rekurziós hivásokra lehetőség. A rekurzió alapvető problémája a visz-

szatérési cimek tárolása és a megfelelő visszatérési cim visszanyerése, másrészt a paraméterek átadása és a lokális változók értékének megőrzése.

A SLIP-rendszerben a visszatérési cimek és a paraméterek a rekurzió folyamán veremben tárolódnak /push-down list/. A SLIP-rendszer inditásakor az INITAS rutin a szabad cellák listájának elkészitése után készit 100 darab üres listafejet W(1)... W(100) nevekkel. Ezeket a listákat nyilvános listáknak nevezzük. A nyilvános listák referencia countja igen magas, hogy a felhasználó véletlenül se tudja elengedni őket. A nyilvános listák neveit a program bármely szegmensében elérhetjük, ha COMMON utasitásban deklaráljuk őket:

```
COMMON /PUBLSS/W (100)
```

A nyilvános listáknak a rekurzió esetén nagy a jelentőségük, a rendszer ezeket a listákat használja veremként a visszatérési cimek, a lokális változók, a praméterek időleges tárolására. A rekurziót a VISIT nevü rutin inditja el. A VISIT egy, a rendszerhez tartozó veremben tárolja a visszatérés cimét /a VISIT hivása alatt soronkövetkező FORTRAN-utasitás cimét/ és a vezérlést átadja a rekurziós utasitásblokk kezdő cimére, amit előzőleg egy egész tipusu változóba egy ASSIGN utasitással kell elhelyeznünk.

Ez a változó a VISIT rutin első bemenő paramétere. A második paramétert a rutin tulajdonképpen nem használja, mivel azonban a rutinba való belépés előtt a paraméterek értéke kiszámitódik, a második paramétert általában a lokális változók megőrzésére használják.

A rekurzió a TERM függvény hivásával zárul be, mely kiveszi a veremből a visszatérés cimét, amelyet a VISIT helyezett el.

Rendelkezésre állnak még további rutinok: PARAMTN rutin a paramétereit a nyilvános listák által reprezentált vermekben helyezi el, a PRESRV (N) az első N nyilvános listát vermeli stb.

4.7 Input-output

Az eredeti SLIP leirásban input-output rutinok nem szerepelnek, hiszen a FORTRAN nyelv minden lehetősége a felhasználó rendelkezésére áll. Kényelmi szempontból célszerünek látszott azonban a rendszerhez illeszteni listastrukturák közvetlen beolvasására, illetve kiirására szolgáló rutinokat.

Az RDLSTA (X) rutin beolvas egy listastrukturát. A teljes

adatlistát és allistáit zárójelek közé kell zárni, a listaelemeket egymástól egy vagy több szóközzel kell elválasztani. Például a (A B (C D) E) jelsorozatból egy A,B,E elemekből álló főlistát és C,D elemekből álló allistát készit a rutin. A függvény értéke, valamint a paraméter értéke a főlista neve lesz.

Listastrukturák kiirására több rutin áll rendelkezésre. A PRLSTA (L) rutin az L nevü listát sornyomtatón, a PULSTA (L) kártyalyukasztón nyomtatja ki a RDLSTA rutinnál leirt formátum szerint.

A PRLSTS (L, N) rutin az L nevü lista elemeit az N paramétertől függően egész vagy lebegőpontos számokként vagy karakter formájában nyomtatja ki.

A listák, illetve allisták kezdetét és végét a nyomtatáskor megjelöli pl.: a RDLSTA tárgyalásánál emlitett lista nyomtatási képe:

```
BEGIN LIST
A
B
     BEGIN SUBLIST
     C
     D
     END SUBLIST
E
END LIST
```

4.8 Egyéb rutinok

Rendelkezésre állnak még rutinok listákkal való különböző müveletek elvégzésére.

Lemásolhatunk egy teljes listastrukturát /LSSCPY (L)/ vagy egy adott listastruktura fő listáját /LSTCPY (L)/, egy adott listához hozzákapcsolhatunk egy másik listát /JOINLS (L1, L2)/, egy adott listát "kiegyenesithetünk", azaz az allistákat a főlistához hozzákapcsolhatjuk /IRONLS (L)/, megállapithatjuk, hogy két listastruktura egyenlő-e /LSTEQL (L1, L2) - két listastruktura egyenlő, ha az adatok egyenlők és a listák strukturája is megegyezik - / stb.

A lista megjelölésére szolgáló LISTMARK feltöltését az MRKLSS (M, LST) függvény végzi el, egy adott lista LISTMARK-ját az LSTMRK (LST) függvény hivásával kaphatjuk meg. A LISTMARK használatának módja a felhasználótól függ.

5. A SLIP SZUBRUTINOK RÉSZLETES LEIRÁSA

5.1 A rendszer előkészitése és inditása

SUBROUTINE INITAS (A, N)

Elkésziti a szabad cellák listáját és a nyilvános listákat.
Praméterek: A a listák számára kijelölt tömb neve
N az A tömb mérete

Az A tömbazonositónak a rutin hivása előtt elő kell fordulnia egy DIMENSION utasitásban. A rutin hivása után a felhasználó nem hivatkozhat többé közvetlenül az A tömbre.

Ha egy FORTRAN-modulban használni kivánjuk a nyilvános listákat, a modulban elő kell fordulnia a COMMON /PUBLSS/W (100) utasitásnak.

A W tömb elemei tartalmazzák a nyilvános listák neveit.

Az INITAS rutint a SLIP-rendszer használata előtt hivni kell.

5.2 Listakészités. Információ elhelyezése és visszanyerése

FUNCTION LIST (K)

Készit egy üres listát /önmagára mutató listafejet/.
Paraméterek: K; bemeneten K = 9 : az uj lista ref. countja nulla lesz;
K ≠ 9 esetén 1.
Kilépéskor K ≠ 9 esetén K a lista neve.
A függvény értéke: a lista neve.

FUNCITON IRALST (L)

Az L nevü lista referencia countját 1-gyel csökkenti. Ha a referencia count értéke 0 lett, a listát elengedi. /Ha az L listának volt leiró listája, akkor a leiró listát is elengedi./
Paraméter: L listanév
A függvény értéke: az L lista uj referencia countja.

FUNCTION MTLIST (L)

Az L nevü listából üres listát készit.

Paraméter: L a kiüritendő lista neve
A függvény értéke: a kiüritett lista neve.

FUNCTION LPURGE (L)

Az L nevü lista allistáit elengedi.
Paraméter: L listanév
A függvény értéke: az elengedett allisták száma.

FUNCTION NEWTOP (D, L)

Az L nevü lista tetején - a listafejtől jobbra - elhelyezi a D adatot. Ha D egy listanév, az L lista tetején listakapcsolót helyez el és a D lista referencia countját 1-gyel megnöveli.
Paraméterek: D az elhelyezni kivánt adat; tipusa tetszőleges,
L annak a listának a neve, melyre az adatot el kivánjuk helyezni,
A függvény értéke: a felfüggesztett uj cella cime.

FUNCTION NEWBOT (D, L)

Az L nevü lista aljára - a listafejtől balra - elhelyezi a D adatot. Müködése NEWTOP-éhoz hasonló.

FUNCITON NXTRGT (D, K)

Egy cimmel megadott listaelem mellé jobbra elhelyezi a D adatot. Ha D egy listanév, kapcsolócellát készit és a D lista referencia countját 1-gyel megnöveli.
Paraméterek: D az elhelyezni kivánt adat,
K-ban annak a listaelemnek a cime van, amely mellé jobbra az P adatot kivánjuk elhelyezni.

FUNCTION NXTLFT (D, K)

Egy cimmel megadott listaelem mellé balra elhelyezi a D adatot. Müködése NXTRGT-éhoz hasonló.

FUNCTION PALIST (A, N)

Egy listát készit 1-es referencia counttal és ezen a listán elhelyezi az A tömb elemeit.
Praméterek: A tömbazonositó
N a tömb elemeinek száma
A függvény értéke: a kapott lista neve.

FUNCTION PASBLS (A, N)

Egy listát készit 0-as referencia counttal és ezen a listán elhelyezi az A tömb elemeit. Müködése PALIST-éhoz hasonló.

FUNCTION SUBST (D, K)

Egy cimmel megadott cella DATUM részét D-re cseréli ki.
Praméterek: D az uj adat
K-ban azon cella cime van, melynek DATUM részét D-re kívánjuk kicserélni.
A függvény értéke: A K-ban tárolt cimü cella régi DATUM része.

FUNCTION SUBSBT (D, L)

Az L nevü lista alján - a listafejtől balra - lévő DATUM részt D-re cseréli ki.
Praméterek: D az uj adat
L listanév
A függvény értéke: Az L nevü lista aljának régi DATUM része.

FUNCTION SUBSTP (D, L)

Az L nevü lista tetején - a listafejtől jobbra - lévő DATUM részt D-re cseréli ki. SUBSBT párja.

FUNCTION BOT (L)

Az L nevü lista aljára elhelyezett DATUMOT szolgáltatja.
Praméter: L listanév
A függvény értéke: Az L nevü lista aljának DATUM része.

FUNCTION TOP (L)

Az L nevü lista tetején elhelyezett DATUMOT szolgáltatja. BOT párja.

FUNCTION POPBOT (L)

Kitörli az L nevü lista alján lévő elemet.
Praméter: L listanév
A függvény értéke: Az L nevü lista aljáról kitörölt elem DATUM része.

FUNCTION POPTOP (L)

Kitörli az L nevü lista tetején lévő elemet. POPBOT párja.

FUNCTION DELETE (K)

A K cimü cellát kitörli.
Paraméter: K-ban a kitörlendő listaelem cimét kell megadni.
A függvény értéke: a kitörölt listaelem DATUM része.

Az eddig felsorolt függvények paraméterének tipusa tetszőleges lehet. A FORTRAN ugyanis csak a paraméter cimét adja át, a paraméter értékét assembly rutinok kezelik.

5.3 Végighaladás a listán, a SEQUENCER és a READER

FUNCTION SEQRDR (L)

Elkésziti az L nevü listához a sequencert.
Paraméter: L listanév
A függvény értéke: a sequencer, amelyet egy FORTRAN változóba kell elhelyezni, pl.:

```
Z = SEQRDR (L)
```

FUNCTION SEQLL (Z, F)

Lineáris sequencer, bal irányban halad a listán soronkövetkező listaelemig.
Paraméter: Z a sequencer
F valós tipusu változó, a következő értékek valamelyikét veszi fel:
0.0 ha kapcsolóhoz érkeztünk,
1.0 ha listafejhez érkeztünk,
-1.0 ha lista elemhez érkeztünk.
A függvény értéke: a listaelem DATUM része.

FUNCTION SEQLR (Z, F)

Lineáris, jobbirányu haladás. SEQLL párja.

FUNCTION SEQSL (Z, F)

Strukturális, balirányu haladás.
Paraméterek: Z a sequencer
F valós tipusu változó, a következő értékek valamelyikét veszi fel:
1.0 ha listafejhez érkeztünk
-1.0 ha listaelemhez érkeztünk.
A függvény értéke: a listaelem DATUM része.

FUNCTION SEQSR (Z, F)

Stukturális, jobbirányu haladás. SEQSL párja.

FUNCTION LRDROV (L)

READER változót készit az L listához.
Praméter: L listanév
A függvény értéke: a reader neve, amelyet egy FORTRAN változóba kell elhelyezni:, pl.:
LR = LRDROV (L)

FUNCTION ADVLEL /ADVLER (LR, F)

Lineáris advancer, bal/jobb irányban halad a listán mindaddig, amig egy listaelemet nem talál /a cella ID része = 0/. Allistákba nem ereszkedik le.
Praméterek: LR a reader,

F valós FORTRAN változó. Értéke 0,0, ha a keresés sikeres volt, azaz listaelemet talált; -1.0, ha nem volt sikeres a keresés, vagyis a lista fejéhez érkezett vissza.

A függvény értéke: a listaelem DATUM része.

FUNCTION ADVLNL/ADVLNR (LR, F)

Lineáris advancer, bal/jobb irányban halad a listán mindaddig, amig egy listakapcsolót nem talál /a cella ID része = 1/. Allistákba nem ereszkedik le. Praméterek és a függvény értéke: mint ADVLEL/ADVLER-nél.

FUNCTION ADVLWL/ADVLWR (LR, F)

Lineáris advancer, bal/jobb irányban halad a listán. A keresés tárgya listafejtől különböző listaelem.
Paraméterek és a függvény értéke: mint ADVLEL/ADVLER-nél.

FUNCTION ADVSEL/ADVSER (LR, F)

Strukturális advancer, bal/jobb irányban halad a listán mindaddig, amig egy listaelemet nem talál /a cella ID része = 0/. Allistákba leszáll; a keresés az egész listastrukturán történik.
Paraméterek : LR a reader

F valós FORTRAN változó. Értéke 0.0, ha a keresés sikeres volt, azaz listaelemet talált; -1.0, ha nem volt sikeres a keresés, vagyis a lista fejéhez érkezett vissza.

A függvény értéke: a cella DATUM része.

FUNCTION ADVSNL/ADVSNR (LR, F)

Strukturális advancer, bal/jobb irányban halad a listán mindaddig, amig listakapcsolót nem talál /a cella ID része = 1/. Allistákba leszáll.

Paraméterek és a függvény értéke: mint ADVSEL/ADVSER-nél.

FUNCTION ADVSWL/ADVSWR (LR, F)

Strukturális advancer, bal/jobb irányban halad a listán. A keresés tárgya: listafejtől különböző listaelem. Allistákba leszállás történik.

Paraméterek és a függvény értéke: mint ADVSEL/ADVSER-nél.

FUNCTION INITRD (LR)

Kezdeti állapotba hozza az LR readert; a reader a listafejre fog mutatni.

Praméter: LR a reader

A függvény értéke: a reader neve.

FUNCTION LCNTR (LR)

Az LR nevü reader szintszámlálóját /level counter/ adja eredményül.

Paraméter: LR a reader

A függvény értéke: a szintszámláló.

FUNCTION LPNTR (LR)

Annak a cellának a cimét adja eredményül, amelyre a reader mutat. A reader változatlan marad.

Paraméter: LR a reader

A függvény értéke: az aktuális cella cime.

FUNCTION LRDRCP (LR)

Lemásolja az LR readerlánc aktuális állapotát.

Paraméter: LR - a reader

A függvény értéke: a lemásolt reader neve.

FUNCTION LVLRVT (LR)

A readert visszaállitja ugy, hogy a listastruktura legfelső szintü listájának azon kapcsoló elemére mutasson, amelyről a leszállás történt.
Paraméter: LR - a reader
A függvény értéke: a reader neve.

FUNCTION LVLRV1 (LR)

A readert visszaállitja ugy, hogy a listastruktura eggyel magasabbszintü listájának azon kapcsolóelemére mutasson, amelyről a leszállás történt.
Paraméter: LR a reader
A függvény értéke: a reader neve.

FUNCTION RDRATN (LR)

A reader megváltoztatása nélkül a lista következő eleméről megállapitja, hogy listakapcsoló-e.
Paraméter: LR - a reader
A függvény értéke: -1.0 ha a következő elem nem kapcsoló
0.0 ha a következő elem kapcsoló.

FUNCTION IRARDR (LR)

Elengedi az LR.nevü readert.
Paraméter: LR a reader
A függvény értéke: a szintszámláló utolsó állapota.

5.4 A leiró lista

FUNCTION LISTAV (L)

Az L nevü listához egy üres leiró listát készit.
Paraméter: L listanév
A függvény értéke: a leiró lista neve.

FUNCTION LDATVL (AT, VL, L)

Az L nevü lista leiró listáján elhelyezi az AT attributumot és a VL értéket. Ha az L listának nincs leiró listája, a függvény készit egy leiró listát és elhelyezi rajta az AT, VAL párt.

Paraméterek: AT az elhelyezni kivánt attributum,
VL az elhelyezni kivánt érték
L listanév

A függvény értéke: a leiró lista neve.

FUNCTION MAKEDL (L, M)

Az L nevü listát az M nevü lista leiró listájává teszi.
Paraméterek: L listanév
M listanév
A függvény értéke: az M lista neve.

FUNCTION NEWVAL (AT, VAL, L)

Az L nevü lista leiró listáján az AT attributumhoz tartozó értéket VAL-ra cseréli ki. Ha a leiró listán az AT attributum nem szerepelt, akkor AT-t elhelyezi a leiró listán a VAL értékkel együtt. Ha az L listának nincs leiró listája, a függvény készit egy leiró listát és elhelyezi rajta az AT, VAL párt.
Paraméterek: AT attribútum,
VAL az elhelyezni kivánt uj érték,
L listanév
A függvény értéke: az AT-hez tartozó régi érték, ha AT szerepelt a leiró listán, különben 0.

FUNCTION ITSVAL (AT, L)

Az L nevü lista leiró listáján az AT attributumhoz tartozó értéket adja eredményül. Ha az L listának nincs leiró listája, hibajelzést ad.
Paraméterek AT attributum
L listanév
A függvény értéke: az AT attributumhoz tartozó érték, ha volt ilyen attributum, különben 0.

FUNCTION NOATVL (AT, L)

Az L nevü lista leiró listájáról kitörli az AT attributumot és a hozzá tartozó értéket.
Paraméterek: AT attributum
L listanév
A függvény értéke: az attributumhoz tartozó érték.

FUNCTION MTDLST (L)

Az L nevü lista leiró listájából üres listát készit, ha volt az L listának leiró listája.
Paraméter: L listanév
A függvény értéke: L lista neve.

FUNCTION IRASDL (L)

Az L nevü lista leiró listájának referencia countját 1-gyel csökkenti. Ha ez 0, a leiró listát elengedi.
Paraméter: L listanév
A függvény értéke: a leiró lista referencia countja, ha volt leiró lista, különben -1.

FUNCTION NAMEDL (L)

Az L nevü lista leiró listájának a nevét adja eredményül.
Paraméter: L listanév
A függvény értéke: a leiró lista neve.

5.5 Rekurzió

FUNCTION PARMT2 (D1, D2)

A D1 ill. D2 adatot elhelyezi a W(1), illetve a W(2) nyilvános lista - mint egy verem - tetején.
Paraméterek: D1 és D2 tetszőleges adat
A függvény értéke: D1

Ha a programozónak több adatot kell rekurzió előtt vermelni, PARMT2-höz hasonló rutint irhat. PARMT2 programja megtalálható 6.-ben.

SUBROUTINE PRESRV (K)

Az első K nyilvános lista legfelső eleméből még egy példányt elhelyez az illető listán.
Paramétere: K egész.

BUBROUTINE RESTOR (K)

Az első K darab nyilvános lista legfelső elemét eltávolitja.
Paramétere: K: egész.

SUBROUTINE RTNDRS (N)

Program előkészitése rekurzióra.
Paramétere: N: egész változó. Értékét a rutin megváltoztatja, de a felhasználó nem használhatja fel.
Rekurzió kezdése előtt egy cimketáblázat kezdőcimét helyezi el N-be a következő utasitássorozat hatására:

```
ASSIGN 8888  TO  J
GOTO  J, (8888)
```

8888 CALL RTNDRS (N)

Az RTNDRS rutin használatát az ICL 1905-ös gép FORTRAN reprezentációja teszi szükségessé. Az egyöntetüség kedvéért a CDC 3300-as gépen is bevezettük.

FUNCTION VISIT (J, N, X)

A rekurzió lehetőségének megteremtésével vezérlésátadást végez a J cimkére.

Paraméterek: J egész változó; tartalma egy cimke, melyet előzőleg ASSIGN utasitással kell belehelyezni, pl: ASSIGN 765 TO J

N egész változó, melynek az RTNDRS rutin ad értéket.

X dummy változó, irható helyébe függvénykifejezés. Pl. PARMT2, amely elvégzi a lokális változók vermelését.

SUBROUTINE TERM (V, X)

A rekurzió megelőző szintjére való visszatérést végzi. A program a legutolsó VISIT-re való hivatkozás utáni helyen folytatódik.

Paraméterek: V az az érték, amelyet a VISIT függvénynek fel kell vennie.

X dummy változó, ha a VISIT-ben X helyén PARMT2 referencia volt, itt a megfelelő RESTOR hivatkozás kell.

5.6 Input-output

FUNCTION RDLSTA (L)

Listastrukturák karakter formában való beolvasására szolgál. Egy lista vagy allista kezdetét egy bal zárójel, a lista vagy allista végét jobb zárójel jelzi. A listaelemeket egymástól szóköz karakterrel kell elválasztani. Egy listaelem maximálisan 8 karakterből állhat. Ha egy elem 8 karakternél kevesebb karakterből áll, a hiányzó karakterek szóköz-karakterrel lesznek kiegészitve. A beolvasott karakterek a sornyomtatón kinyomtatásra kerülnek.

Paraméter: L: az elkészitett lista nevét veszi fel értékül.

A függvény értéke: az elkészitett lista neve.

FUNCTION PRLSTA (L)

Az L nevü listát karakterformában kinyomtatja a sornyomtatón. A nyomtatási kép megegyezik az RDLSTA által kivánt formával.

Paraméter: L a kinyomtatni kivánt lista neve

A függvény értéke: a lista neve.

FUNCTION PULSTA (L)

Az L nevü listát kártya/papirszalag lyukasztón kinyomtatja karakter formában. PRLSTA párja.

FUNCTION PRLSTS (L, K)

Az L nevü listát a K paramétertől függő formában sornyomtatón kinyomtatja, a listák és allisták kezdetét, végét a nyomtatáskor megjelöli.

Paraméterek: L a kinyomtatandó lista neve,
K a nyomtatás formátumát határozza meg:
K=1 esetben I14; egész formában
K=2 esetben A8; karakter formában
K=3 Flo.4; lebegőpontos formában történik a listaadatok kiirása.

A függvény értéke: listanév.

SUBROUTINE ACTIVE az adott pillanatban aktiv cellákat a sornyomtatón kinyomtatja.

5.7 Listák megjelölése

FUNCTION MRKLSS (M, L)

Az L nevü lista és a hozzá tartozó összes allista LISTMARK-jába az M értéket teszi.

Paraméterek: M egész konstans; $0 \leq M < 64$
L a főlista neve,

A függvény értéke megegyezik L-lel.

FUNCTION MRKLST (M, L)

Az L nevü lista LISTMARK-jába az M értéket teszi.

Paraméterek: M egész konstans; $0 \leq M < 64$
L listanév

A függvény értéke megegyezik L-lel.

FUNCTION LSTMRK (L)

Az L nevü lista LISTMARK-ját adja.

Paraméter: L listanév

A függvény értéke: a LISTMARK.

5.8 Lista müveletek

FUNCTION GROUPS (L, N)

Az L lista legfelső szintjén lévő adatokat az L listáról leakasztja és N darabonként egy-egy allistára helyezi el. Az igy nyert allistákat az eredeti listára visszaakasztja.
Paraméterek: L listanév
N egész változó
A függvény értéke: listanév

FUNCTION INLSTL (L, K)

Az L nevü lista összes elemét lekapcsolja és egy másik listán lévő K cimü cellától balra elhelyezi. Az L lista üres lista lesz.
Paraméterek: L listanév
K változó, melyben egy cella cime van.
A függvény értéke: az üressé vált lista neve.

FUNCTION INLSTR (L, K)

Az L nevü lista összes elemét lekapcsolja és egy másik listán lévő K cimü cellától jobbra elhelyezi. Az L lista üres lista lesz. INLSTL párja.

FUNCTION IRONLS (L)

"Kiegyenesiti" az L nevü listát, ugyhogy az L lista minden eleme a legfelső szintre kerül; allista nem lesz.
Paraméter: L listanév
A függvény értéke: listanév

FUNCTION JOINLS (L1, L2) az L2 lista elemeit az L1 lista aljára akasztja, az L2 lista fejét pedig elengedi.
Paraméter: L1 listanév
L2 listanév
A függvény értéke: L1 listanév

FUNCTION NULSTL/NULSTR (K, L)

Készit egy uj listát és erre az uj listára felakasztja az L nevü lista K cimü elemétől balra/jobbra elhelyezkedő összes elemét /a K cimü cellát is beleértve/. Az L nevü listán csak a K cimü cellától jobbra/balra lévő elemek maradnak.
Paraméterek: K változó, egy cella cimét tartalmazza
L listanév

A függvény értéke: az uj lista neve.

FUNCTION LSTCPY (L)

Az L nevü lista legfelső szintjét lemásolja egy uj listába. Az L lista változatlan marad. Az allistákról nem készül másolat.
Paraméterek: L listanév
A függvény értéke: az uj lista neve.

FUNCTION LSSCPY (L)

Az L nevü listát az összes allistáival együtt lemásolja. Az L nevü lista változatlan marad.
Paraméterek: L listanév
A függvény értéke: az uj lista neve.

5.9 Listavizsgáló rutinok

FUNCTION LSTEQL (L1, L2) összehasonlitja az L1 és L2 listát. Két lista azonos, ha a lista minden eleme azonos, és a listák felépitése is azonos.
Paraméterek: L1 listanév
L2 listanév
A függvény értéke: 0 ha a két lista azonos,
-1 különben.

FUNCTION LISTMT (L) megvizsgálja, hogy az L nevü lista üres-e, azaz hogy a lista csak a listafejből áll-e.
Paraméter: L listanév
A függvény értéke: 0 ha a lista üres
-1 különben.

FUNCTION MEMBER (X, L) megvizsgálja, hogy az X változó szerepel-e az L nevü listán, mint adat.
Paraméterek: X adat
L listanév
A függvény értéke: az első cella cime L-en, amelyben
X az adat. Ha ilyen cella nincs, a függvény értéke: -1.

FUNCTION NELEML (L)

Az L nevü lista legfelső szintjén lévő listaelemek számát adja eredményül /kapcsolók nélkül/.
Paraméterek: L listanév
A függvény értéke: a listaelemek száma.

FUNCTION NELEMS (L)

Az L nevü listastruktura elemeinek a számát adja eredményül /kapcsolók nélkül/.

Paraméter: L listanév

A függvény értéke: a listaelemek száma.

FUNCTION NNAMEL (L)

A L nevü lista legfelső szintjén lévő listakapcsolók számát adja eredményül.

Paraméter: L listanév

A függvény értéke: a listakapcsolók száma.

FUNCTION NNAMES (L)

Az L nevü listastruktura kapcsolóinak számát adja eredményül.

Paraméter: L listanév

A függvény értéke: a listakapcsolók száma.

FUNCTION NWORDL (L)

Az L nevü lista legfelső szintjén lévő listaelemek és listakapcsolók számát adja eredményül.

Paraméterek: L listanév

A függvény értéke: az elemek és listakapcsolók száma.

FUNCTION NWORDS (L)

Az L nevü listastruktura elemeinek és listakapcsolóinak számát adja eredményül.

Paraméterek: L listanév

A függvény értéke: az elemek és listakapcsolók száma.

5.10. Vegyes rutinok

FUNCTION INTGER (X)

Egy valós változó tartalmát helyezi el egész változóban - értékkonverzió nélkül /assembly rutin/.

Paraméter: X valós

A függvény értéke: megegyezik X tartalmával.

Megjegyzés: Az L = INTGER (X) utasitás helyettesithető CALL STRDIR (X, L)-lel.

FUNCTION EQUAL (A, B)

Változók tartalmának összehasonlitása /assembly rutin/.

Paraméterek: A és B tetszőleges tipusu változók

A függvény értéke: 0.0, ha A és B tartalma egyező,
≠ 0.0, ha A és B különbözőek.

FUNCTION MADNBT (L, K)

Az L nevü lista balról /aljáról/ számitott K-dik elemének cimét adja.

Paraméter: L listanév
K egész

A függvény értéke: a K-dik elem cime.

FUNCTION MADNTP (L, K)

Az L nevü lista jobbról (tetejéről) számitott K-adik elemének cimét adja. MADNBT párja.

A következő rutinok közvetlen hivására a felhasználónak általában nincs szüksége, a rendszer belső rutinjai, az előzőekben ismertetett rutinok hivatkoznak rájuk. (A rendszert alaposan ismerő felhasználó azonban - kellő körültekintéssel - közvetlenül is hivatkozhat rájuk.)

FUNCTION MADHDR (L)

Az L nevü lista fejének a cimét adja eredményül.

Paraméter: L listanév

A függvény értéke: a listafej cime.

FUNCTION MADRGT (K)/ MADLFT (K)

A K-ban tárolt cimü cella után jobbra/balra következő cella gépi cimét adja.

Paraméter: K egy cella cime

A függ.ény értéke: a megfelelő szomszédos cella gépi cime.

FUNCTION MADATR (AT, L)

Az L nevü lista leiró listáján az AT attributum cimét adja.

Paraméterek: L listanév
AT attributum

A fuggvény értéke: az AT attributum gépi cime, ha ilyen attributum volt;
különben -1.

FUNCTION MHDRDL (L)

Az L nevü lista leiró listájának gépi cimét adja.

Paraméter: L listanév

A függvény értéke : a leiró lista fejének cime.

FUNCTION NUCELL (X)

A szabad cellák listájáról leakaszt egy cellát. /Megvizsgálja, hogy a kapott cella listakapcsoló-e. Ha a leakasztott cella listakapcsoló, annak a listának a referencia countját 1-el csökkenti, melyre a listakapcsoló hivatkozik. Ha a referencia count 0 lesz, akkor elengedi a listát.
Paraméter: X nem játszik szerepet.
A függvény értéke: az uj cella cime.

FUNCTION NAMTST (L)

Ellenőrzi, hogy L listanév-e.
Paraméter: L listanév
A függvény értéke: 0, ha L listanév
-1, ha L nem listanév

FUNCTION NMIFDL (L, J, N)

Az L nevü lista leiró listájának nevét adja eredményül, ha létezett leiró lista, különben a J változóban tárolt cimkére történik vezérlésátadás. A függvény hivása előtt J-nek egy ASSIGN utasitásban, N-nek egy CALL RTNDRS /N/ utasitásban kell szerepelnie.
Paraméterek: J, N egész változók
L listanév
A függvény értéke: a leiró lista fejének a cime.

FUNCTION REED (K)

Az advancer rutinok belső rutinja. A K-ban tárolt cimü cella DATUM részét adja.
Paraméter: K gépi cimet tartalmaz
A függvény értéke: a cella DATUM része.

FUNCTION LOFRDR (LR)

A reader LOFRDR mezőjének tartalmát adja, azaz az aktuális lista nevét.

Paraméter: LR reader neve
A függvény értéke: az aktuális lista neve.

SUBROUTINE RCELL (K)

A szabad cellák listájára akasztja a K-ban tárolt cimü cellát.
Paraméter: K cella cimet tartalmaz.

FUNCTION SETNAM (K, L)

Az L változóban az M-ben tárolt cimet helyezi el, ugy hogy L egy listanév legyen.
Paraméter: M-ben listafej cime
L változó, kimenő paraméter
A függvény értéke: azonos L-el.

FUNCTION LENG (X)

Az adott gépen a cella méretét adja meg;
értéke két szomszédos cella gépi cime közötti különbség.
Paraméter: X: a rutin nem használja
A függvény értéke: az ICT 1905 és a CDC 3300 gépen 4.

FUNCTION MAX (X)

Az elérhető maximális méret az adott számológépen.
Paraméter: X: a rutin nem használja
A függvény értéke: ICT 1905 gépen 32000.

FUNCTION MATUM (K)

A K-ban megadott cimü cella DATUM részét adja eredményül.
Paraméter: X egy cella gépi cimét tartalmazza
A függvény értéke: a K-ban tárolt cimü cella DATUM része.

FUNCTION DATUM (K)

A K-ban megadott cimü cella DATUM részét adja eredményül.
Paraméter: K egy cella gépi cimét tartalmazza.
A függvény értéke: a K-ban tárolt cimü cella DATUM része.

FUNCTION DSETIN (I, LL, LR, K)

A K-ban tárolt cimü cella DATUM részének ID, LNKL, LNKR mezőit tölti ki. Ha az első három paraméter valamelyike - 1, azt a részt változatlanul hagyja
Paraméterek: I az ID rész
LL az LNKL rész
LR az LNKR rész
K a cella gépi cimét tartalmazza,
A függvény értéke: a kitöltött cella DATUM része.

FUNCTION DSTRIN (M, K)

A K-ban tárolt cimü cella DATUM részébe az M értéket helyezi el.
Paraméter: M tartalmazza az elhelyezni kivánt értéket.
A függvény értéke: M

FUNCTION ID (C)

A C ID részét adja. /assembly rutin/.
Paraméter: C egy cella CONT része/!/
A függvény értéke: a C ID része.

FUNCTION CONT (K) / INHALT (K)

A K-ban tárolt cimü cella CONT részét adja /assembly rutin/.
Paraméter: K-ban egy cella cime van
A függvény értéke: CONT rész.

FUNCTION LNKL (C) LNKR (C)

A C LNKL/LNKR részét adja /assembly rutin/.
Paraméter: C egy cella CONT része
A függvény értéke: LNKL/LNKR mező értéke.

FUNCTION MADOV (A)

Az A változó gépi cimét adja /assembly rutin/.
Paraméter: A: FORTRAN változó
A függvény értéke: A gépi cime.

FUNCTION STRIND (A, K)

A K-ban tárolt cimre A tartalmát teszi /assembly rutin/.
Paraméter: A változó
K-ban egy cella cime van.
A függvény értéke: A

FUNCTION STRDIR (A, K)

A tartalmát K-ba teszi /assembly rutin/.
Paraméter: A változó
K változó
A függvény értéke: A

FUNCTION SETIND (I, L, R, K)

A K-ban tárolt cimü cella ID, LNKL, LNKR mezőjében elhelyezi az I, L, R paraméterek értékét.
Ha ID, L, R valamelyike - 1, az a rész változatlan marad /assembly rutin/.
Paraméterek: I a cella ID része
L a cella LNKL része
R a cella LNKR része
K-ban a cella cime van.
A függvény értéke: a cella CONT része.

FUNCTION SETDIR (I, L, R, C)

A C ID, LNKL, LNKR mezőjébe az I, L, R, C, értékeket helyezi el. A - 1 értékü paraméternek megfelelő rész változatlan marad /assembly rutin/.
Paraméterek: I a cella ID része
L a cella LNKL része
R a cella LNKR része
C változó
A függvény értéke: a cella /C/ CONT része.

FUNCTION LANORM (W)

W tartalmát mindaddig balra shifteli, mig a legkevésbé szignifikáns karakter nem egy betüköz karakter /assembly rutin/.
A függvény értéke megegyezik W-vel.

SUBROUTINE RTNDRS (N) / GOBACK (J, N)

Az ASSIGN utasitásban szereplő változó, J által reprezentált cim megkeresésére szolgálnak. Az adott FORTRAN compilertől függ, hogy az ASSIGN utasitásban szereplő cimkéket hogyan kezeli. Ha a kijelölt cimkékhez táblázatot készit, az RTNDRS rutin a táblázat cimét adja eredményül az N változóban. Ha a compiler nem használ táblázatot, az RTNDRS rutinra nincs szükség. /Mivel azonban a FORTRAN nyelven irt rutinokban szerepel, minden esetben léteznie kell az RTNDRS rutinnak is!/. A GOBACK rutin a vezérlést a J-ben /előzőleg ASSIGN utasitásban kell szerepelnie!/ kijelölt cimre adja át.

Ha a compiler nem használ táblázatot, akkor a GOBACK rutin N paraméterére sincs szükség, az egyöntetüség kedvéért azonban mindig kell használni /assembly rutinok/!

SUBROUTINE RETMAD (Z)

A VISIT assembly nyelven irt segédrutinja. Z-ben a VISIT hivása után következő utasitás cimét szolgáltatja.

FUNCTION SQOUT (F, S)

Karakter manipulációs rutin. Az ICT 1900-as és CDC 3300-as gépen az S-ben tárolt 8 karakterből az F maszk segitségével kiemel egy karaktert és ezt adja a függvényérték legalsó karakterpoziciójában /assembly rutin/.

Paraméterek: F karakter maszk
S nyolc karaktert tartalmazó szó.

A függvény értéke: a kiemelt karakter.

FUNCTION SQUIN (F, S, V)

Karaktermanipulációs rutin. Az S-ben tárolt legalsó karaktert beilleszti V-nek az F maszk által meghatározott karakterpoziciójára /assembly rutin/.

Paraméterek: F és S mint SQOUT-nál
V nyolc karaktert tartalmazó szó, ahova egy karaktert a rutin beilleszt.

A függvény értéke: V uj értékével egyezik meg.

FUNCTION SHIN (N, S, V)

Karaktermanipulációs rutin. V tartalmát N bittel balra lépteti és S legalsó karakterét V végére illeszti /assembly rutin/.

Paraméterek: N egész; = 6, 12, 18, ...
S és V szöveget tároló változók

A függvény értéke: V uj értéke.

6. PÉLDÁK

6.1 A PARMT3 rutin **paramétereit** elhelyezi az első három nyilvános lista tetején:

```
SUBROUTINE PARMT3 (D1, D2, D3)
COMMON/PUBLSS/W (100)
CALL NEWTOP (D1, W (1))
CALL NEWTOP (D2, W (2))
CALL NEWTOP (D3, W (3))
PARMT3 = D1
RETURN
END
```

6.2 A következő programrészlet megszámolja a Q nevü lista allistáinak számát

```
1 |  SL =  SEQRDR (Q)
2 |  J  =  0
3 |  CALL SEQLR (SL, F)
4 |  IF (F) 3,5,7
5 |  J = J + 1
6 |  GO TO 3
7 |  .
  |  .
  |  .
```

Az 1 utasitás létrehozza a Q lista sequencerét, melyet a 3 utasitásban használunk. A 3 utasitás helyett irhattuk volna

```
CALL SEQLL (SL, F)
```

utasitást is. Ha F > 0, visszaérkeztünk a listafejhez.

6.3 Összegezzük az L listán lévő adatokat; feltéve, hogy azok valós számok

```
     SL = SEQRDR (L)
     SUM = 0.0
2    X = SEQLL (SL, F)
```

```
      IF (F) 1,2,3
1     SUM = SUM + X
      GO TO 2
3     .
      .
      .
```

Ha az adatok egész számok, a következőképpen járhatunk el:

```
      .
      .
      EQUIVALENCE (X, M)
      .
      .
      .
      SL = SEQRDR  (L)
      MSUM = 0
2     X = SEQLL (SL, F)
      IF (F) 1,2,3
1     MSUM = MSUM + M
      GO TO 2
3     .
      .
      .
```

6.4 A következő programrészlet 10-ig a páros számokat elhelyezi az L2 listán, a páratlan számokat az L1 listán

```
      .
      .
      DIMENSION AREA (500)
      .
      .
      .
      CALL INITAS (AREA, 500)
      .
      .
      .
      L1 = LIST (L1)
      L2 = LIST (L2)
      .
      .
      DO  3  I = 1,10
      IF (I - I/2 * 2) 2,1,2
```

```
1     CALL NEWBOT (I, L2)
      GO TO 3
2     CALL NEWBOT (I, L1)
3     CONTINUE
C     MOST AZ L1 ES L2 LISTAKBOL OSSZEALLITJUK
C     10-IG AZ EGESZ SZAMOK LISTAJÁT AZ L-ET
      L = LIST  (L)
      DO 7 I = 1,5
4     B = POPBOT (L1)
5     CALL NEWBOT (B, L)
6     B = POPBOT (L2)
7     CALL NEWBOT (B, L)
C     L LISTA AZ EGESZ SZAMOK LISTAJA
      .
      .
      .
```

A 4 és 5 utasitást összevontan is irhattuk volna:

```
      CALL NEWBOT (POPBOT (L1), L)
```

ugyanigy a 6 és 7 utasitás:

```
      CALL NEWBOT (POPBOT (L2), L)
```

6.5 Példa a rekurzióra: készitsünk függvényt az n! kiszámitására!

```
      FUNCTION NFACT (N)
      COMMON /PUBLSS/W (100)
1     ASSIGN 5 TO K
2     CALL RTNDRS (M)
3     NFACT = INTGER (VISIT (K, M, PARMT2 (N, 0)))
4     RETURN
5     IF (EQUAL (1, TOP (W (1)))) 6,9,6
6     C = VISIT (K, M, PARMT2 (INTGER (TOP (W (1)))-1,0))
7     NF = NF * INTGER (TOP (W(1)))
8     CALL TERM (NF, RESTOR (2))
9     NF = 1
10    CALL TERM (NF, RESTOR (2))
      END
```

A következő táblázat az NFACT függvényt analizálja utasitásról utasitásra N = 3 esetben.

Az utasitás előtt	a rekurzió szintje	W(1) nyilvános lista állapota	a visszatérési cimek listája	NF	NFACT
3	0	üres	üres	-	-
5	1	3	4	-	-
6	1	3	4	-	-
5	2	2,3	7,4	-	-
6	2	2,3	7,4	-	-
5	3	1,2,3	7,7,4	-	-
9	3	1,2,3	7,7,4	-	-
10	3	1,2,3	7,7,4	1	-
7	2	2,3	7,4	1	-
8	2	2,3	7,4	2	-
7	1	3	4	2	-
8	1	3	4	6	-
4	0	üres	üres	6	6

6.6 Tekintsünk egy N1 csomópontból álló hálózatot. A csomópontokhoz és a csomópontokat összekötő utakhoz számértékeket rendelünk. A csomópontokhoz hozzárendeljük a csomópont sorszámát, mig az élékhez valamilyen más értéket. A következő program kinyomtatja a hálózat első és utolsó pontját összekötő összes lehetséges utat és minden uthoz tartozó élek adatának összegét.

```
      PROGRAM NETWORK
      DIMENSION SPACE (2000), L(50), IPATH (50)
1     CALL INITAS (SPACE, 2000)
2     READ (5,99) N1, N2
C     N1 a CSOMOPONTOK, N2 AZ UTAK SZAMA
3     DO  4  I = 1, N1
4     CALLNEWTOP (I, LIST (L (I)))
C     L (I) AZ I-DIK CSOMOPONT LISTAJA, A CSOMOPONT
C     SORSZAMAT TARTALMAZZA ADATKENT
5     DO 8 J = 1, N2
6     READ (5,99) NSTART, NFIN, ISCORE
7     CALL NEWBOT (L (NFIN), L (NSTART))
8     CALL NEWVAL (NFIN, ISCORE, L (NSTART))
```

```
C       MINDEN L (I) LISTAN ELHELYEZZUK AZON LISTAK
C       NEVEIT MELYEKBE AZ I CSOMOPONTROL EL LEHET JUTNI
C       AZ L (J) LISTA LEIROLISTAJAN AZ UTHOZ RENDELT
C       SZAMERTEKET HELYEZZUK EL
 9      M = LRDROV (L (1))
12      NODE = INTGER (ADVSER (M, F))
13      IF (F) 23,14,23
14      M1 = LCNTR (M) + 1
15      IPATH (M1) = NODE
16      IF (NODE - N1) 12,17,12
C       MEGTORTENT EGY LEHETSEGES UT CENERALASA
17      K = 0
18      DO 20 KK = 2, M1
19      I = IPATH (KK - 1)
        WRITE (6, 101) I
20      K = K + ITSVAL (IPATH (KK), L (I))
        WRITE (6, 100) K
C       EGY UT ERINTETT CSOMOPONTJAINAK ES
C       AZ UTHOZ TARTOZO ERTEK KINYOMTATASE

21      GO TO 12
23      CALL IRARDR (M)
24      STOP
99      FORMAT (3 I2)
100     FORMAT (/// 10 H K ERTEKE = , I3)
101     FORMAT (6 H NODE = , I2)
        END
```

7. A RENDSZER HASZNÁLATÁVAL KAPCSOLATOS TUDNIVALÓK

Egy SLIP cella mérete általában két FORTRAN "tárolóegység". A CDC 3300 és ICL 1905 gépeken ez 4 egymásutáni szót jelent. Egy programban a rendelkezésre álló szabad cellák száma, ha az INITAS rutin paraméterenként megadott tömb méretét N-nel jelöljük: (2 x N-400)/4.

Mivel a rendszer egy FORTRAN változót a tipusától függetlenül mindig két szóként értelmez, a rendszert használó programot mindig olyan módban kell forditani, hogy az INTEGER tipusu változók a valós tipusuakkal egyforma hosszuak legyenek.

Segédtároló kezelésére a rendszer nem nyujt lehetőséget.

A SLIP-rendszer input/output rutinjai meghatározott logikai perifériaszámot használnak. A 99-es logikai perifériaszám az inputot, a 100-as logikai perifériaszám az outputot jelenti. A felhasználó a rendszeren kivül is használhatja ezeket a logikai perifériaszámokat.

A rendszer a következő hibaüzeneteket adhatja:

AN ATTEMPT HAS BEEN MADE TO DELETE A HEADER üzenetet a DELETE függvény adja. Jelentése a DELETE függvény leirásánál. Nem fatális hiba, a program futása folytatódik.

A LIST WAS REQUIRED AS AN OPERAND BUT WAS NOT FOUND üzenetet a NAMTST függvény adja, ha listára nem listanévvel hivatkoztunk. Fatális hiba, a program megáll, tovább nem folytatható.

THE LIST OF AVAILABLE SPACE IS EXHAUSTED üzenetet a NUCELL függvény adja. A program megáll, tovább nem folytatható.
A hibaüzenetek a 100-as számu periférián jelennek meg.

A SLIP-rendszer a FORTRAN nyelv speciális szubrutincsomaggal való kiterjesztését jelenti. Ezért mindkét gépen készült egy speciális könyvtár, amely a SLIP-rendszer szubrutinjait tartalmazza.

7.1. A SLIP-rendszer használata az ICL 1905-ös gépen

Az ICL 1905-ös gépen a SLIP-rendszert használó programokat a

"SLIP LIBRARY" könyvtárszalag segitségével kell leforditani, amely az XFAM FORTRAN fordítóprogramon és a FORTRAN szubrutin-blokkon kivül a SLIP rutincsomagot is tartalmazza. Ha tehát a programozó használni kivánja a SLIP-rendszert, a programkisérő lapon fel kell tüntetnie, hogy a program forditását a SLIP-LIBRARY-val kéri.

A programleiró szegmensben a rendszerben használt logikai periféria számokhoz perifériát kell rendelni, például:

```
INPUT 99 = TRO
OUTPUT 100 = LPO
```

A programot tilos COMPRESS módban forditani!

A SLIP-rendszert használó program nyomkövetése nagy körültekintést igényel, ugyanis akár a felhasználó által használt, akár a rendszer által használt rekurziós rutinok megváltoztathatják a nyomlistát.

7.2. A SLIP-rendszer használata a CDC-3300-as gépen.

A SLIP-rendszer szubrutincsomagjából egy kisegitő library készült. A CDC-3300-as gépen müködő MASTER operációs rendszernek megfelelően a SLIP-rendszert használó programnak a kisegitő library-t meg kell nyitni; a következő vezérlő kártyáknak kell az input deck-ben szerepelni;

```
$**DEF(0,,ALIB,MASTER,SLIP-LIBRARY,00,A***2,I)
$**DEF(0,,ADIR,MASTER,SLIP-LIBRARY-DIRECTORY,,00,A***2,I)
```

A FORTRAN-program leforditása után a betöltéskor a SLIP-rutinokat a már megnyitott segéd libraryból a

```
$AUX,ALIB,ADIR
```

vezérlőkártyával kell betölteni.

A rendszerben szereplő logikai periféria számokat

a

```
$FILE,99=INP és a
$FILE,100=OUT
```

vezérlőkártyákkal kell kijelölni.

Példa:

```
$JOB, . . .
$SCHED, . . .
$*DEF(O,,ALIB,MASTER,SLIP-LIBRARY,OO,A**2,I)
$*DEF(O,,ADIR,MASTER,SLIP-LIBRARY-DIRECTORY,OO,A**2,I)
$FILE,99=INP
$FILE,100=OUT
$FTNU (. . .)
       . . .
       FORTRAN deck
       . . .
       FINISH
$X,LGO
$AUX,ALIB,ADIR
     adatok
     .
     .
     .
77
88
```

7.3. Alapvető SLIP

A SLIP-szubrutinok nagy száma a jobb idegzetü programozókat is megijesztheti. Ezért ebben a részben felsoroljuk azokat a rutinokat, amelyekkel a rendszer megismerését célszerü kezdeni.

INITAS (A,N)	:	a rendszer megnyitása,
RDLSTA (L)	:	lista beolvasása,
PRLSTA (L)	:	lista kinyomtatása,
LIST (L)	:	egy uj lista létrehozása,
MTLIST (L)	:	egy lista celláinak törlése,
DELETE (K)	:	adott cella törlése egy listáról,
NEWBOT (D,L)	:	uj cellaelem a lista tetejére,
NEWTOP (D,L)	:	uj cellaelem a lista aljára,
NXTLFT (D,K)	:	uj cellaelem egy elemtől balra,
NXTRGT (D,K)	:	uj cellaelem egy elemtől jobbra,
BOT (L)	:	egy lista legfelső adata
TOP (L)	:	egy lista legalsó adata,
SUBST (D,K)	:	egy cella tartalmának helyettesitése,

SEQRDR (L) : a listán végighaladó szekvencer létrehozása,
SEQLL (S,F) : végighaladás a listán (lineáris, balirányu)
SEQLR (S,F) : végighaladás a listán, lineáris, jobbirányu
SEQSL (S,F) : végighaladás a listán, strukturális, balirányu
SEQSR (S,F) : végighaladás a listán, strukturális, jobbirányu
LRDROV (L) : readerkészités
ADVabc (L,F): advancerek.

* * * *

8. A SLIP-RUTINOK ÖSSZEFOGLALÓ TÁBLÁZATA

NÉV	HIVÁS	OLDAL
ACTIVE	CALL ACTIVE	28
ADVabc	W=ADVabc(LR,F)	22, 23, 14
BOT	W=BOT (L)	20, 11
CONT	C=CONT (K)	36
DATUM	D=DATUM (K)	35
DELETE	W=DELETE (K)	21
DSETIN	D=DSETIN (I,LL,LR,K)	35
DSTRIN	D=DSTRIN (M,K)	36
EQUAL	W=EQUAL (A,B)	32
GOBACK	CALL GOBACK (J,N)	37
GROUPS	W=GROUPS (L,N)	29
ID	I=ID (C)	36
INHALT	J=INHALT (K)	36
INITAS	CALL INITAS (A,N)	18, 9, 16
INITRD	LR2=INITRD (LR)	23
INLSTL	L1 = INLSTL (L,K)	29
INLSTR	L1 = INLSTR (L,K)	29
INTGER	J = INTGER (X)	32, 8
IRALST	M = IRALST (L)	18, 10
IRARDR	M = IRARDR (LR)	24
IRASDL	L1= IRASDL (L)	26
IRONLS	L1= IRONLS (L)	29, 17
ITSVAL	M = ITSVAL (AT,L)	25, 15
JOINLS	L = JOINLS (L1, L2)	29, 17
LANORM	J = LANORM (W)	37

NÉV	HIVÁS	OLDAL
MHDRDL	M = MHDRDL (L)	33
MRKLSS	L1 = MRKLSS (M,L)	29, 17
MRKLST	L1 = MRKLST (M,L)	28
MTDLST	L1 = MTDLST (L)	25
MTLIST	L1 = MTLIST (L)	18
NAMTST	J = NAMTST (L)	34
NAMEDL	L1 = NAMEDL (L)	26
NELEML	M = NELEML (L)	30
NELEMS	M = NELEMS (L)	31
NEWBOT	K = NEWBOT (D,L)	19, 11
NEWTOP	K = NEWTOP (D,L)	19, 11
NEWVAL	M = NEWVAL (AT, VAL, L)	25, 15
NMIFDL	K = NMIFDL (L,J,N)	34
NNAMEL	M = NNAMEL (L)	31
NNAMES	M = NNAMES (L)	31
NOATVL	M = NOATVL (AT,L)	25
NUCELL	K = NUCELL (X)	34
NULSTL	L1 = NULSTL (K, L)	29
NULSTR	L1 = NULSTR (K, L)	29
NWORDL	M = NWORDL (L)	31
NWORDS	M = NWORDS (L)	31
NXTLFT	M = NXTLFT (D, K)	19, 11
NXTRGT	M = NEXTRGT (D,K)	19, 11
PALIST	W = PALIST (A,N)	19
PARMT 2	B = PARMT 2 (D1, D2)	26, 16
PASBLS	W = PASBLS (A, N)	20
POPBOT	W = POPBOT (L)	20, 11
POPTOP	W = POPTOP (L)	20, 11
PRESERV	CALL PRESERV (K)	26, 16
PRLSTA	W = PRLSTA (L)	27, 17

NÉV	HIVÁS	OLDAL
PRLSTS	W = PRLSTA (L, N)	28, 17
PULSTA	W = PULSTA (L)	28, 17
RCELL	CALL RCELL (K)	34
RDLSTA	W = RDLSTA (L)	27, 16, 17
RDRATN	F = RDRATN (LR)	24
REED	D = REED (K)	34
RESTOR	CALL RESTOR (K)	26
RETMAD	CALL RETMAD (Z)	38
RTNDRS	CALL RTNDRS (N)	26
SEQLL	W = SEQLL (Z, F)	21, 8
SEQLR	W = SEQLR (Z, F)	21, 12
SEQRDR	Z = SEQRDR (L)	21, 12
SEQSL	W = SEQSL (Z, F)	21, 12
SEQSR	W = SEQSR (Z, F)	22, 12
SETDIR	C = SETDIR (I,L,R,C)	37
SETIND	C = SETIND (I,L,R,K)	37
SETNAM	S = SETNAM (K, L)	35
SHIN	V = SHIN (N, S, V)	38
SQOUT	V = SQOUT (F, S)	38
SQUIN	V = SQUIN (F, S, V)	38
STRDIR	A1 = STRDIR (A, K)	36
STRIND	A1 = STRIND (A, K)	36
SUBST	W = SUBST (D, K)	20, 11
SUBSBT	W = SUBSBT (D, L)	20, 11
SUBSTP	W = SUBSTP (D, L)	20, 11
TERM	W = TERM (V, X)	27, 16
TOP	W = TOP (L)	20, 11
VISIT	V = VISIT (J,N, X)	27, 16

IRODALOMJEGYZÉK

[1] WEIZENBAUM, J.: Symmetric List Processor. Communications of ACM Vol 6 /1963/. No 9. PP 524-544.

[2] SMITH, D,K.: An Introduction to the List Processing Language SLIP. Programming Systems and Languages /ed.S.Rosen, 1967./.

[3] RAPHAEL,B. - BOBOROW,D.G. - FEIN,L. - YOUNG,J,W.: A Brief Survey of Computer Languages for Symbolic and Algebraic Manipulation. Symbol Manipulation Languages am Techniques. Proc. of IFIP. conf. on Symbol Manipulation /1967/

[4] BALL,W.E - BERNS,R.I.: AUTOMAST - Automatic Mathematical Analysis and Symbolic Translation.Communications of ACM Vol 9 /1966/ No.8. PP 626-633.

[5] BARBIERI, R.: Computer List - Processing Languages IBM, Data System Division, Ponghkeepsie, New York Technical Report, No. TR00.1209. /1964/

[6] BOBROW,D.G.: - WEIZENBAUM,J.: List Processing and Extension of Language Facility by imbedding.IEEE Transactions on Electronic Computers. Vol.EC-13. /1964/ No.4. pp. 395-400.

[7] COLLINS, G.E.: REFCO III - A Reference Count List Processing System for the IBM 7094. IBM.Watson Res.Center, Yorktown Heights,NY.Res.Report RC-1436 /1965/.

[8] LAPIDUS,A - GOLDSTEIN,M.: Some Experiments in Algebraic Manipulation by Computer. CACM Vol 8. /1965/ No.8.

[9] LAPIDUS,A - GOLDSTEIN,M - GREENSPAN,S.: An Algebraic Manipulator Using Slip. Counant Institute of Math. Sci., New York University /1966/.

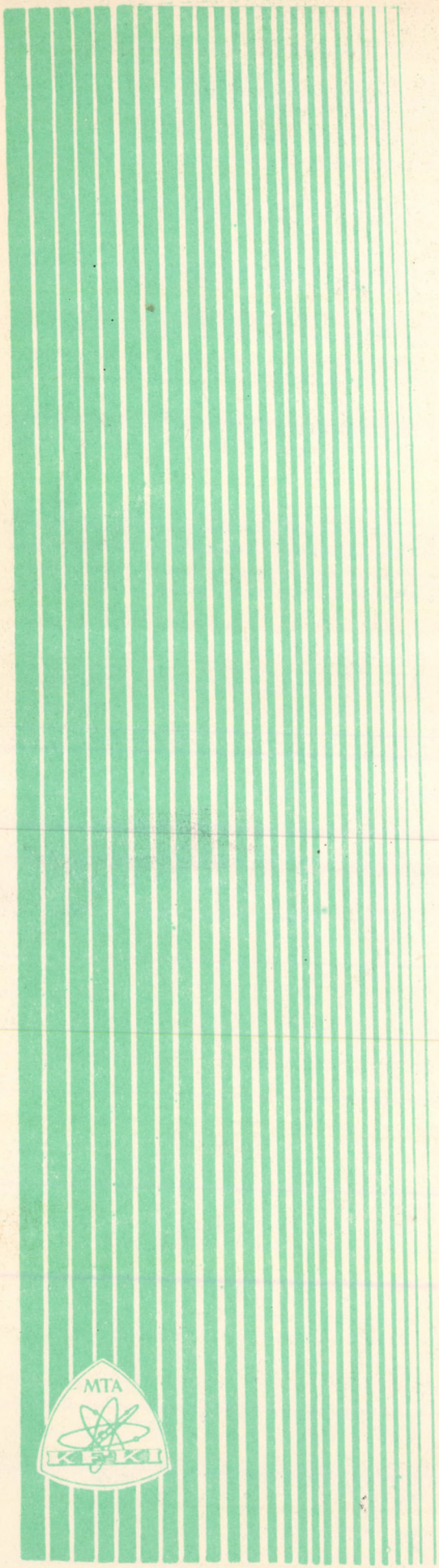
[10] GOTTLIEB,C.C. - NOVAK,R.J.: ALGEM - An Algebraic Manipulater University of Toronto. /1966/.

[11] RAMANI,S.: SLIP Operations on Trees and their Relevance to Problems of Linguistic Interest. Symbol Manipulation Languages and Techniques. Proc.of.IFIP Conf. on Symbol Manipulation /1967/.

[12] Proc.of.the International Joint Conference on Artificial Intelligence. Washington /1969/.

[13] FINDLER N.V. - Me KINZIE W.R.: On a new Tool in Artificial Intelligence Research. An Associate Memory, Parallel Processing /12/ pp. 259-270.

[14] ABELSON,R.P. - REICH,C.M.: Implication Moleculas: A Method for Extracting Meaning from Input Sentences. /12./ pp. 641-647.



Kiadja a Központi Fizikai Kutató Intézet
Felelős kiadó: Varga László a Számitástechnikai Tudományos Tanács elnöke
Szakmai lektor: Ivanyos Lajosné
Példányszám: 300 Törzsszám: 72-6717
Készült a KFKI sokszorositó üzemében, Budapest
1972. május hó